

## **AMENDMENTS TO THE CLAIMS**

This listing of claims will replace all prior versions, and listings, of claims in the application:

### **Listing of Claims:**

- 1           1. (Currently amended) A method to automate isolation of native code  
2     within a computer program that has been compiled to a platform-independent  
3     code, the method comprising:  
4           receiving a library containing a native code sub-routine, wherein the native  
5     code sub-routine provides a service to the computer program;  
6           analyzing the library to determine a defined symbol name for the native  
7     code sub-routine;  
8           creating a proxy sub-routine for the native code sub-routine, wherein the  
9     proxy sub-routine forms a link to the native code sub-routine;  
10          placing the proxy sub-routine into a new library using the defined symbol  
11     name of the native code sub-routine as a symbol name for the proxy sub-routine;  
12          running the native code sub-routine in a first process;  
13          executing the platform-independent code in a second process; and  
14          invoking the native code sub-routine in the first process by ~~calling~~ using  
15     an interprocess call to call the proxy sub-routine from the platform-independent  
16     code in the second process;  
17          wherein operations in the first process are isolated from memory and other  
18     system resources belonging to the second process by ensuring that interactions  
19     between the first process and the second process take place through the  
20     interprocess call, so that an error in the first process does not, one of, corrupt

21 | memory belonging to the second process or interfere with the second process in  
22 | any other way.

1           2. (Original) The method of claim 1, further comprising:  
2           providing a proxy platform-independent native interface (PINI) to the  
3           library containing the native code sub-routine; and  
4           transparently transforming local PINI calls into calls to the proxy PINI,  
5           wherein transforming local PINI calls into calls to the proxy PINI  
6           preserves an original control flow, and  
7           wherein upcalls and downcalls are guaranteed to be executed by a same  
8           thread of a process that executes the native code sub-routine.

1           3. (Original) The method of claim 1, wherein analyzing the library to  
2           determine the defined symbol name includes analyzing the library to determine  
3           call arguments for the defined symbol name.

1           4. (Original) The method of claim 3, wherein analyzing the library to  
2           determine call arguments for the defined symbol name is accomplished at runtime  
3           by analyzing a current call frame.

1           5. (Original) The method of claim 3, further comprising copying call  
2           arguments from the proxy sub-routine to a call to the native code sub-routine.

1           6. (Original) The method of claim 3, further comprising returning a result  
2           value from the native code sub-routine to the proxy sub-routine.

1           7 (Canceled).

1           8. (Original) The method of claim 1, wherein the proxy sub-routine and  
2 the native code sub-routine communicate through inter-process communication.

1           9. (Original) The method of claim 1, wherein forming the link to the native  
2 code sub-routine includes translating a data element from a first address width in  
3 the computer program to a second address width in the native code sub-routine.

1           10. (Currently amended) A computer-readable storage medium storing  
2 instructions that when executed by a computer cause the computer to perform a  
3 method to facilitate automated isolation of native code within a computer program  
4 that has been compiled to a platform-independent code, the method comprising:  
5           receiving a library containing a native code sub-routine, wherein the native  
6 code sub-routine provides a service to the computer program;  
7           analyzing the library to determine a defined symbol name for the native  
8 code sub-routine;  
9           creating a proxy sub-routine for the native code sub-routine, wherein the  
10 proxy sub-routine forms a link to the native code sub-routine;  
11           placing the proxy sub-routine into a new library using the defined symbol  
12 name of the native code sub-routine as a symbol name for the proxy sub-routine;  
13           running the native code sub-routine in a first process;  
14           executing the platform-independent code in a second process; and  
15           invoking the native code sub-routine in the first process by ~~calling using~~  
16 an interprocess call to call the proxy sub-routine from the platform-independent  
17 code in the second process;  
18           wherein operations in the first process are isolated from memory and other  
19 system resources belonging to the second process by ensuring that interactions  
20 between the first process and the second process take place through the  
21 interprocess call, so that an error in the first process does not, one of, corrupt

22 | memory belonging to the second process or interfere with the second process in  
23 | any other way.

1           11. (Original) The computer-readable storage medium of claim 10, the  
2 method further comprising:  
3           providing a proxy platform-independent native interface (PINI) to the  
4 library containing the native code sub-routine; and  
5           transparently transforming local PINI calls into calls to the proxy PINI,  
6           wherein transforming local PINI calls into calls to the proxy PINI  
7 preserves an original control flow, and  
8           wherein upcalls and downcalls are guaranteed to be executed by a same  
9 thread of a process that executes the native code sub-routine.

1           12. (Original) The computer-readable storage medium of claim 10,  
2 wherein analyzing the library to determine the defined symbol name includes  
3 analyzing the library to determine call arguments for the defined symbol name.

1           13. (Original) The computer-readable storage medium of claim 12,  
2 wherein analyzing the library to determine call arguments for the defined symbol  
3 name is accomplished at runtime by analyzing a current call frame.

1           14. (Original) The computer-readable storage medium of claim 12, the  
2 method further comprising copying call arguments from the proxy sub-routine to a  
3 call to the native code sub-routine.

1           15. (Original) The computer-readable storage medium of claim 12, the  
2 method further comprising returning a result value from the native code sub-  
3 routine to the proxy sub-routine.

1           16 (Canceled).

1           17. (Original) The computer-readable storage medium of claim 10,  
2 wherein the proxy sub-routine and the native code sub-routine communicate  
3 through inter-process communication.

1           18. (Original) The computer-readable storage medium of claim 10,  
2 wherein forming the link to the native code sub-routine includes translating a data  
3 element from a first address width in the computer program to a second address  
4 width in the native code sub-routine.

1           19. (Currently amended) An apparatus that facilitates automated isolation  
2 of native code within a computer program that has been compiled to a platform-  
3 independent code, the apparatus comprising:

4           a receiving mechanism that is configured to receive a library containing a  
5 native code sub-routine, wherein the native code sub-routine provides a service to  
6 the computer program;

7           an analyzing mechanism that is configured to analyze the library to  
8 determine a defined symbol name for the native code sub-routine;

9           a creating mechanism that is configured to create a proxy sub-routine for  
10 the native code sub-routine, wherein the proxy sub-routine forms a link to the  
11 native code sub-routine;

12          a placing mechanism that is configured to place the proxy sub-routine into  
13 a new library using the defined symbol name of the native code sub-routine as a  
14 symbol name for the proxy sub-routine;

15          a running mechanism that is configured to run the native code sub-routine  
16 in a first process;

17 an executing mechanism that is configured to execute the platform-  
18 independent code in a second process; and  
19 an invoking mechanism that is configured to invoke the native code sub-  
20 routine in the first process by ealling-using an interprocess call to call the proxy  
21 sub-routine from the platform-independent code in the second process;  
22 wherein operations in the first process are isolated from memory and other  
23 system resources belonging to the second process by ensuring that interactions  
24 between the first process and the second process take place through the  
25 interprocess call, so that an error in the first process does not, one of, corrupt  
26 memory belonging to the second process or interfere with the second process in  
27 any other way.

1 20. (Original) The apparatus of claim 19, further comprising:  
2 a providing mechanism configured to provide a proxy platform-  
3 independent native interface (PINI) to the library containing the native code sub-  
4 routine; and  
5 a transforming mechanism that is configured to transparently transform  
6 local PINI calls into calls to the proxy PINI,  
7 wherein transforming local PINI calls into calls to the proxy PINI  
8 preserves an original control flow, and  
9 wherein upcalls and downcalls are guaranteed to be executed by a same  
10 thread of a process that executes the native code sub-routine..

1 21. (Original) The apparatus of claim 19, wherein the analyzing  
2 mechanism is further configured to analyze the library to determine call arguments  
3 for the defined symbol name.

1           22. (Original) The apparatus of claim 21, wherein analyzing the library to  
2 determine call arguments for the defined symbol name is accomplished at runtime  
3 by analyzing a current call frame.

1           23. (Original) The apparatus of claim 21, further comprising a copying  
2 mechanism that is configured to copy call arguments from the proxy sub-routine  
3 to a call to the native code sub-routine.

1           24. (Original) The apparatus of claim 21, further comprising a returning  
2 mechanism that is configured to return a result value from the native code sub-  
3 routine to the proxy sub-routine.

1           25 (Canceled).

1           26. (Original) The apparatus of claim 19, further comprising an inter-  
2 process communication mechanism that is configured so that the proxy sub-  
3 routine and the native code sub-routine can communicate.

1           27. (Original) The apparatus of claim 19, further comprising an address  
2 width translating mechanism that is configured to translate an address from a first  
3 address width in the computer program to a second address width in the native  
4 code sub-routine.